The Surrogate Data Type in TSQL2

September 16, 1994

A TSQL2 Commentary

The TSQL2 Language Design Committee

Title	The Surrogate Data Type in TSQL2
Primary Author(s)	Christian S. Jensen and Richard T. Snodgrass
Publication History	September 1994. TSQL2 Commentary.

TSQL2 Language Design Committee

Richard T. Snodgrass, Chair	University of Arizona
rts@cs.arizona.edu	Tucson, AZ
Ilsoo Ahn	AT&T Bell Laboratories
ahn@cbnmva.att.com	Columbus, OH
Gad Ariav	Tel Aviv University
ariavg@ccmail.gsm.uci.edu	Tel Aviv, Israel
Don Batory	University of Texas
dsb@cs.utexas.edu	Austin, TX
James Clifford	New York University
jcliffor@is-4.stern.nyu.edu	New York, NY
Curtis E. Dyreson	University of Arizona
curtis@cs.arizona.edu	Tucson, AZ
Ramez Elmasri	University of Texas
elmasri@cse.uta.edu	Arlington, TX
Fabio Grandi	Universitá di Bologna
fabio@deis64.cineca.it	Bologna, Italy
Christian S. Jensen	Aalborg University
csj@iesd.auc.dk	Aalborg, Denmark
Wolfgang Käfer	Daimler Benz
kaefer%fuzi.uucp@germany.eu.net	Ulm, Germany
Nick Kline	University of Arizona
kline@cs.arizona.edu	Tucson, AZ
Krishna Kulkarni	Tandem Computers
kulkarni_krishna@tandem.com	Cupertino, CA
T. Y. Cliff Leung	Data Base Technology Institute, IBM
cleung@vnet.ibm.com	San Jose, CA
Nikos Lorentzos	Agricultural University of Athens
eliop@isosun.ariadne-t.gr	Athens, Greece
John F. Roddick	University of South Australia
roddick@unisa.edu.au	The Levels, South Australia
Arie Segev	University of California
segev@csr.lbl.gov	Berkeley, CA
Michael D. Soo	University of Arizona
soo@cs.arizona.edu	Tucson, AZ
Suryanarayana M. Sripada	European Computer-Industry Research Centre
sripada@ecrc.de	Munich, Germany

Abstract

This document proposes syntax and informal semantics for the inclusion of a `SURROGATE` data type in the TSQL2 query language.

1 Introduction

The `SURROGATE` data type should, without compromizing the special properties of surrogates, be treated similarly to how all other data types are treated in SQL2. Additional goals include that the extension should be as minimal as possible. For example, as few reserved words as possible should be introduced. Also, the extension should be consistent and compatible with the syntax for user-defined time.

2 Informal Definition

Surrogates are unique identifiers that can be compared for equality, but the values of which cannot be seen by the users. In this sense, a surrogate is "pure" identity and does not describe a property (i.e., it has no observable value).

For this reason, a `SURROGATE` data type cannot be treated identically to how other data types are treated. For example, at tuple cannot be asigned a specific value for a `SURROGATE` attribute. Rather, the user must indicate that the value to be assigned must be a new value never used before, or, alternatively, must indicate some `SURROGATE` attribute of some tuple that the assigned value must be identical to.

For example, if surrogates are used for the identification of employees and a new tuple with information about an existing employee is to be entered then it is specified that the surrogate of the new tuple is to be identical to that of the existing tuple(s) for the particular employee.

Beyond the semantics just mentioned, it is the responsibility of the user to give meaning to surrogates. For example, the user may, or may not, use attributes of `SURROGATE` type as keys.

The `SURROGATE` data type is treated like any other data type when creating and altering relation schemas. An example follows.

```
CREATE TABLE Employee (Name CHAR, Id SURROGATE, Dept CHAR, Salary INT)
                AS VALID STATE
```

Due to the special semantics of surrogates (e.g., values of surrogate attributes cannot be seen), update is special. With the relation instance just defined, this is an example of an insertion of a new employee.

```
INSERT INTO r VALUES ('Ben', NEW, 'Toy', 30) VALID PERIOD '1 Jan 1993 - 31 Mar 1993'
```

The reserved word `NEW` indicates that the system must supply a new surrogate value that has never before been used in the database. Thus, `NEW` is not a particular surrogate, but may be thought of as a variable that is instantiated by the system, when the insertion takes place, to a surrogate that has never been used before. For example,

```
INSERT INTO r VALUES ('Ben', NEW, 'Toy', 30) VALID PERIOD '1 Jan 1993 - 31 Mar 1993'
INSERT INTO r VALUES ('Bill', NEW, 'Toy', 30) VALID PERIOD '1 Apr 1993 - 30 Jun 1993'
```

results in two tuples, with distinct surrogates, being appended to relation `r`; further, the two surrogates are distinct from all other surrogates that have been used.

The next example illustrates how new information may be linked to existing information by means of surrogates. In this example, we represent individual employees by means of the surrogate-valued attribute, `Id`.

```
INSERT INTO Employee
     SELECT ('Benjamin', Employee.Id, 'Toy', 30)
     VALID PERIOD '1 Apr 1993 - 31 May 1993')
     FROM Employee
     WHERE Employee.Name = 'Ben' AND
          Employee OVERLAPS PERIOD '5 Jan 1993 - 10 Jan 1993'
```

Here, we add more information for the same person (who changed name). We say that the `Id` of the new tuple should be that of the tuple with the `Name` value Ben some time during the specified time interval in January 1993, assuming that we know that only one person was named Ben between January 5 and January 10, 1993.

Like attributes of other data types, attributes of type `SURROGATE` are allowed to have null values. However, like attributes of other data types, constraints such as `NOT NULL` may also be imposed on attributes of `SURROGATE` type.

When considering the querying of relations with surrogate-valued attributes, the semantics of surrogates again have some implications. Specifically, since surrogate values cannot be viewed by the user (including application programs), an error results when a surrogate attribute is included in an outer-most target list (i.e., the outer-most `SELECT` clause) of a query.

Further, the use of "$*$" in the `SELECT` clause when argument relations contain surrogate-valued attributes needs special attention. Consider an example where we retrieve all information for employees in the `Toy` department. We would like to for,mulate the query as follows.

```
SELECT *
FROM Employee
WHERE Employee.Dept = 'Toy'
```

To make this possible, we adopt the convention that the "$*$" does not select surrogate valued attributes. Thus `SELECT *` in the query above is equivalent to selecting all attributes in `Employee`, with the exception of attributes of type `SURROGATE`. As a result, the semantics of surrogates as well as the usability of the "$*$" notation is retained. In order to retrieve a surrogate attribute (in a subquery), the attribute must be referenced explicitly.

Next, only equality comparison is defined for surrogates. Thus, surrogates may be tested for equality (and not-equal, using, `=` and `NOT` or `<>`), but an error results when an attempt is made to apply other (e.g., "greater-than") predicates to surrogates.

Surrogates do not replace keys, but rather supplement them. While surrogates may be used for connecting information within the database, values of surrogates have no real-world meaning. Keys, on the other hand, may be used for relating information in the database with real-world entities. In the sample table `Employee`, the attribute `Name` may be declared as a key (e.g., `PRIMARY KEY ( Name )`). It is via `Name`, rather than via `Id`, that the user establishes the connection between a tuple and the actual employee the tuple is about.

Acknowledgements

This work was supported in part by NSF grants ISI-8902707 and ISI-9302244, IBM contract #1124 and the AT&T Foundation. Christian S. Jensen was also supported in part by the Danish Natural Science Research Council, grants no. 11–1089–1 SE and no. 11–0061–1 SE.

A Modified Language Syntax

The organization of this section follows that of the SQL2 document. The syntax is listed under corresponding section numbers in the SQL2 document. All new or modified syntax rules are marked with a bullet ("•") on the left side of the production.

Where appropriate, we provide disambiguating rules to describe additional syntactic and semantic restrictions. We assume that the reader is familiar with the SQL2 proposal, and that a copy of the proposal is available for reference.

A.1 Section 5.2 <token>

Two reserved words were added.

<reserved word> ::=
• | `NEW`
• | `SURROGATE`

A.2 Section 6.1 <data type>

<data type> ::=
• | <surrogate type>

<surrogate type> ::=
- SURROGATE

Additional general rules:

1. Values of type SURROGATE cannot be seen (displayed). Consequently, attributes of SURROGATE type are not allowed in the outermost SELECT clause of a query. Also, attributes of surrogate type cannot be assigned an explicit value.
2. A special reserved word, NEW may be used when updating an attribute value of SURROGATE type. The new value is a previously unused value.
3. Values of type SURROGATE can only be compared with respect to identity.

A.3 Section 13.8 <insert statement>

The <insert column list> is modified to permit the use of the NEW reserved word.

<insert column list> ::=
- <insert column> [{ <comma> <insert column> }...]

The <insert column> is a new production.

<insert column> ::=
- <column name>
- | NEW

Additional general rules:

1. NEW is permitted only when the <data type> of the corresponding column is SURROGATE.